INFLUENCE OF TASK TYPE ON THE RELATION BETWEEN COMMUNICATION AND PERFORMANCE: THE CASE OF SOFTWARE DEVELOPMENT

Oscar Hauptman

90s: 85-013

Working Paper

Management in the 1990s

Massachusetts Institute of Technology
Sloan School of Management

INFLUENCE OF TASK TYPE ON THE RELATION BETWEEN COMMUNICATION AND PERFORMANCE: THE CASE OF SOFTWARE DEVELOPMENT

Oscar Hauptman

90s: 85-013

May 1985

Sloan WP #1712-85

Management in the 1990s
Sloan School of Management
Massachusetts Institute of Technology

INFLUENCE OF TASK TYPE ON THE RELATION BETWEEN COMMUNICATION AND PERFORMANCE: THE CASE OF SOFTWARE DEVELOPMENT

Oscar Hauptman

Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

May 31, 1985

To be presented at the "Managing R&D Effectively" Conference
Manchester Business School, Manchester, UK,
July 10-12, 1985

INFLUENCE OF TASK TYPE ON THE RELATION BETWEEN COMMUNICATION AND PERFORMANCE: THE CASE OF SOFTWARE DEVELOPMENT[1,2,3]

Oscar Hauptman

Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

The relations between communication patterns and performance of software development projects mostly resemble those of technical services, and not development projects, in hardware R&D. The local focus of software development projects in their information requirements is emphasized by the positive influence only of the informal and mostly internal literature, while external contacts, participation in conferences, and formal and external literature were inconsequential. The implications are two-fold: a)on the conceptual level they suggest that a trade-off between coordination and innovation requirements of the task might be an important determinant of optimal communication patterns; b)on the practical level it suggests that "software development" consists of "software engineering" and "software production". Consequently, it should be recognized that as such, these activities should be managed differently - the former as R&D, the later as manufacturing.

Introduction

Coordination versus innovation in software development

Coordination of task activities comes through as a key requirement in what is usually defined as "software development". The seminal "The Mythical Man-Month" (Brooks, 1982) brings up the counter-intuitive notion that "Adding manpower to a late software project makes it later" (p.25). The reasons given by Brooks are that the coordinating requirements of the project, which increase as a second power of the number of project members, substantially dilute the contribution of added manpower. A significant part of it goes to initial learning and integration of the new member into the project team. Another component is the continuous, day-to-day coordination of individual's work with the activities of other team members.

The importance of coordination requirements in software is emphasized

---

1 The research presented in this paper has been sponsored by the Mangement in the Nineties Research Program, Sloan School of Management, MIT.

2 This study has been facilitated by an exceptional level of support by ICL, UK. My gratitude to the Applied Systems members whose continuous participation in this study made it possible. Special acknowledgement to Hugh MacDonald (Technical Directorate), Asa Lanum (Director Applied Systems) and Ken Bodenham (C&TS) of ICL for their unwavering support.

3 My gratitude to Tom Allen whose review and critique significantly contributed to the quality of this study. Finally, Michael Scott-Morton's help was invaluable in launching this research.

by the time allocation of the individual programmer. McCabe's data (1978) indicate that 50% of a typical programmer's time is spent interacting with other team members, while 30% is spent working alone; 20% are non-productive, spent on travel and administration. The recent comprehensive System Dynamics modeling of the software development process (Abdel-Hamid, 1984) describes this phenomenon as "communication losses" (p.183). Compared with the base-line of a single person development "team", communication with other team members represents a coordinating overhead. Using Brooks's formulation (see also Mills, 1976; Scott and Simmons, 1975) it might constitute up to 50% of the total available "man-months" of a thirty members team. This overhead consists of both verbal coordinating interactions, and of increased work-load, due to the necessary interfaces between software modules produced by each individual; they have to be agreed upon, formalized through design, executed in software code, and finally, documented.

A simplified model (Howlett, 1985), which assumes constant coordinating requirements per dyad, and a totally divisible task, examplifies this issue. The net useful output of the team can be computed as:

$$W(n) = n - kn(n-1) \qquad (1)$$

when n is the number of team members, and k is the constant proportion of time consumed by the coordination oriented communication effort. The kn(1-n) component is the coordinating cost of the project. On these premises adding another person will result in a net contribution which can be computed by:

$$W(n+1) - W(n) = [(1+k)(n+1) - k(n+1)^2] - [(1+k)n - kn^2] =$$
$$= (1+k) - k(2n+1) = 1 + k - 2kn - k = \underline{1-2kn} \qquad (2)$$

The "break-even" point of the net contribution, at which it equals zero, is in the realistic range of values when k=5%, and n=10. On these premises, communication is regarded only a costly overhead. The contention that "the time which a group spends on communication is a non-productive time" (Somerville, 1982; p.248) is widely accepted by software management academics. It sounds quite axiomatic.

In comparison with this articulate treatment of planning, control and coordination issues in the software management related literature, most of it fails to address the value of communication in contributing to project performance through technological innovation. The emphasis of Dijkstra (1976) on task discipline "in order to achieve a more reliable and well-understood product" emphasizes the formal and structured administration of software development activities. On the other hand, references to software related technological innovation are few. For instance Boehm (1980) argues that technology transfer to the industrial organization constitutes a major problem in this field. He advocates better training of software professionals in the intricacies of real-life software development, emphasizing the fact that there is much more to software than just programming. He also addresses the issue of the pace of technological innovation in software related core technologies by estimating their "half-lives". Most of them have comparatively long half-lives ranging between 10 and 40 years, with the exception of hardware based technologies such as microprocessors, with a half-life of approximately two years. But even in his statement the technological

issues do not seem to be quite as salient as the coordinative ones: fully 55% and 37% of government acquisition reports identified poor planning and poor control (respectively) as the problem (p.50), in comparison with 18% identifying technology related factors. Freeman (1980) emphasized the special role of the designer in software technology, and advocated the development and implementation of advanced design tools.

This one-sided approach is puzzling in view of the well articulated concepts and empirical evidence from the hardware R&D environment provided by Allen and associates. It is comprehensively summarized in "Managing the Flow of Technology" (Allen, 1977). The studies in the frame of this paradigm reliably showed the usefulness of internal and external communication in contributing to project performance. The two central premises on which the theory of communication in R&D is based are: first, the typical R&D task is sufficiently complex, and requires such a wide assortment of technical expertise that its successful completion depends on adequate information flow from sources external to the task team. Second, because the typical R&D task depends on rapidly evolving technologies, the task team cannot maintain adequate, up to date technical information, especially in projects of long duration. Both emphasize the need for continuous inflow of technical information, usually through informal channels. In addition, the inter-organizational flow through "gatekeepers" also contributes to project's success (Allen and Cohen, 1969).

This controversy can be summarized in the following lines: on the one hand, software development practitioners and researchers consider the coordination to be the most salient factor in determining software development effectiveness. Their field experience suggests that proper planning, management and control, structure and formal rules and regulations will decide project's success or failure. This trend of thought represents the managerial philosophy in the field. On the other hand, in view of the image of software development as an unstructured and intellectually abstract activity, of complexity resembling hardware R&D, Allen's premises are expected to operate in this environment as well. Consequently, we would expect innovation carrying communication to contribute to software development effectiveness.

The process of software development

Before trying to resolve this controversy it is important to consider the idiosyncracies of software development. There is considerable consensus about the list of activities that constitute the software development process, and their temporal sequence. Almost every publication in this area lists in similar vocabulary the typical components of the process as: a)Requirements and specifications, b)Program design, c) Programming, d)Verification and validation, and e)Maintenance (Boehm, 1979; p.54). Somerville's "Software Engineering" (1982) list of contents is almost identical: 2.Requirements, 3. Design, 4.Implemetation: The Programming Language, 5. Implementation II: Programming Practice, 6. Testing and Debugging, 7. Documentation and Maintenance (p. ix-x).

The nature of these activities can be briefly explicated in the following product development scenario: The first stage is the generation of an idea for a new application or a new product, usually through an interaction between marketing experts and designers from the development unit. The second step is the translation of product specification into technical terms by designers and analysts. They resemble construction architects, and this stage is usually described as program architecture.

Next, programmers take over, focusing on detailed design and "code-cutting" - the actual programming. It is usually performed interactively and iteratively with detailed design. If the overall program designer is the "architect", the programmers are the "civil engineers" and "draftsmen". The result of programmers' work is the prototype. After several iterations with the quality assurance department, which is "policing" the reliability of the program, this prototype actually constitutes the final product! Here, in contrast with other industries, from consumer electronics to military hardware, manufacturing or reproduction of the software development prototype is a trivial procedure of simple copying from various memory devices, disks and tapes, to ROMs and floppy diskettes. The value added in this process is negligible, comparable to packaging and documentation and probably lower than the contribution of marketing. But the differences do not end there; in contrast to the "charter" of the R&D staff in hardware industries, the software development team is also responsible for maintenance and support of the final product with the end user. I have witnessed the emotion-loaded reaction of a software development manager to a "Red Alert" telex from the field salesman. It explicitly spelt trouble - "bugs" in the the program, and the development manager had to assume personal responsibility for their occurance and correction. This situation is not atypical in view of the literature in this area (e.g., Boehm, 1979; Somerville, 1982)

In the light of the above special attributes of the software development process, the over-emphasis of coordination might be well grounded. Still it is interesting to know whether the innovation carrying communication is inconsequential for this type of task.

The nature of the task as a determinant of the relation between communication and performance

Pelz and Andrews (1966) classification of R&D task types suggests that software develoment is neither basic nor applied research. The process is well structured, and is based on specific routines of design, production, and testing. It is obviously not "Work of general nature intended to apply to a broad range of application or to the devlopment of a new knowledge about an area" (Allen, Lee and Tushman, 1980: p.3). On the other hand, in view of its task attributes it can be either "The application of known facts and theory to solve a particular problem through exploratory study, design, and testing of new components or systems" (development) or "Cost/performance improvements to existing products, processes, or systems. Recombination, modification, and testing of systems using existing knowledge. Opening new markets for existing products" (technical services) (there, p.4). Allen, et al. (1980) suggest the that impact of project team communication patterns on its effectiveness and performance is not omnipresent, neither uniform. In their conclusions they emphasize that:

> "Many have long realized the difference between university basic research and engineering in industrial laboratories. Now it appears that the distinctions that must be made are even more subtle than that: staff performing more reserach-oriented tasks in an industrial laboratory will behave very differently and have quite different needs from staff concerned with product and process development. These in turn are quite different from those concerned with product modification and adaptation" (p.11).

According to their findings, while development projects benefited

from communication with the rest of the R&D laboratory and the rest of the firm, especially marketing and production, research and technical service projects did not follow this pattern. In addition, only the technical service projects consistenly benefited from management controlled communication with the environment external to the project team (see Allen et al., 1980, Table VI, IX, XII-XIV).

These results might have interesting implications in addition to those suggested by Allen, et al. While it is very plausible that the nature of the task determines the informational needs of the project team, the usual organizational demands for coordination and control can be as salient in more structured and routine tasks. It is reasonable to assume that the tasks for which coordination is more important might have lower needs for informally acquired external technical information. This inverse relation actually implies a situation of trade-off between the two types of communication - the coordination-oriented versus the one which enhances technology transfer and innovation.

The nature of software development provides an exciting setting to address the above questions: it is a well structured type of task, for which coordination seems essential. It would be interesting to compare the relations between communication and performance in software with Allen et al. (1980) results for hardware R&D for two reasons: first, similarity with relations found for either research, development, or technical service will help to classify software development along the hardware R&D continuum. In addition, because coordination is considered to be important for software development, the nature of the relations between communication and performance will provide a "measuring stick" for tasks which require more coordinative than innovation carrying communication.

Research Setting and Methods

The study was carried out at the application software development division of International Computers Limited (ICL), UK. The division is responsible for development and support of a variety of software product -lines, from language compilers and software super-structures to self-contained software packages for end users. These product are marketed all over the world to a variety of customers, from large public bureaucracies (turn-key and custom-made) to small firms. The technological assortment consists of knowledge engineering, language compilers, relational systems, data dictionary, graphics, and recently CAD and artificial intelligence. Most of the products share the core technology of software development. This technology, being approximately twenty years old, is presently reaching maturity.

The facilities of the division are spread among several geographic sites in the UK. Reading, Berkshire includes the divisional headquarters and another two sites. Groups of 35 to 150 employees are located in Manchester, Kidsgrove, and Bracknell. Smaller groups of one to ten employees are spread in London, Leeds, Slough, Birmingham and Newcastle.

At the time of the study the Applied Systems division employed approximately 650 people, some of them part-time freelancers. The study included mostly technical professionals, marketing experts and various operational and technical consultants (n=503). The division was organized into three Business Centers and four Sectors, not including administration. These units were sub-divided into departments which contained between one and three software development projects. The projects were comparatively stable work areas.

Internal communication

Communication data were collected via questionnaires distributed in August-September 1984. The participants were asked to indicate against the names of all the members of survey population the most recent date at which a work-related communication took place. The results are preliminary because the data of October and December 1984, and May 1985 surveys are not included here.

The response rate was approximately 60%, which were sufficient to generate at least unilateral nominations of communication partners for 95% of the population.

Aggregate measures of average communication intensity and its coefficient of variation for project members with progressively larger and mutually exclusive units were computed similarly to Allen, Lee & Tushman (1980).

1. Intraproject communication: Communication with all the members of one's immediate project team.

2. Intradepartmental communication: Communication with all the members of the department. A department was usually based either on a core technology, e.g. graphics, or focused on a specific market segment. This aggregate does not have an analogue in Allen, et al. (1980) study.

3. Communication with the rest of the division: This was divided similarly to Allen et al. into: 1)Intra Business Center/Sector; and 2)Inter Business Center/Sector. Business Centers/Sectors usually address a larger market segment, e.g. Public Administration which includes projects in legal and health data base markets.

The intradepartmental and intra Business Center/Sector communications were eventually divided into communication with a)programmers, b)designers, and c)managers.

Other sources of information

Literature and external contacts were found useful to development project performance (Allen, 1977; Allen, Tushman, & Lee, 1979; Allen, Lee, & Tushman, 1980). The following data, standardized for project size, were collected via questionnaires to test the importance of these sources for software development:

1. Frequency of readership of informal, and mostly internal publications such as internal ICL reports, software and hardware manuals, and informal technical reports from universities, government and other firms.

2. Frequency of readership of formal, and external publications such as scientific and professional journals, conference proceedings, trade periodicals, and textbooks.

3. Frequency of communication with suppliers, customers, consultants, university staff, staff of other firms, and finally friends, was aggregated into an overall index of external communication of the project.

4. Participation in external conferences by project members between 1982 and 1984.

Project performance

Project performance was evaluated on several criteria, suggested by most of the twenty managers in Applied Systems inteviewed for this purpose. These criteria were similar with those mentioned by numerous publications in software engineering or software development area.

The criteria that were selected are of two types: 1)organizational or managerial, which emphasize the coordinative dimension of the task -

success in meeting project designated budget and schedule, and 2)technological, which emphasize the innovative dimension - success in meeting project specifications of functionality, and the comparative quality of the product in reliability, maintainability and flexibility. An overall measure of project success was also collected.

The project evaluations were collected separately from a variety of evaluators in Applied Systems. They typically included the project manager, the department manager, and the marketing, quality assurance, and structure and design experts. The number of evaluators was between three and six, which is considered both sufficient and cost-efficient (Libby and Blashfield, 1978). The reliability of the additive scales for each criteria was assessed by computing its Cronbach alpha. The alphas for most of the criteria, with the exception of maintainability and flexibiltity, were adequately high (between 0.71 and 0.87), justifying the use of aggregate additive scales based on the scores of individual evaluators.

An important factor, which ensures the direction of the causal link from communication to performance, is the 5 months lag between the collection of these data; performance evaluations were collected in February 1985.

It should be mentioned here that the high correlation (Pearson r between 0.78 and 0.92) among the various performance criteria prevent separate analysis of organizational (coordination oriented), and technological (innovation oriented) criteria

Results: Relationship between Communication and Performance

Communication within the project

Similar to Allen et al. (1980), the frequency of intra-project communication seems inconsequential as far as performance is concerned (Table 1). In addition, the variation of intra-project communication

Table 1: Relation Between Performance and Communication Among Project Team Members

Results from Allen, Lee and Tushman (1980)[2]		Results from the Software Environment (present study)[1]				
		Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research	0.05	0.07	-0.16	-0.18	-0.07	-0.19
Development	0.14					
Technical Service	0.17					

1-Kendall's Tau. 2-Pearson Correlation. See footnote for sample sizes.

across project members is similarly not important, though they resemble development most than technical services in view of Allen et al. data (Table 2).

Sample sizes for all the tables are: 1)Allen et al. - a)research - 13, b)development - 21, technical service - 15; 2)Present study - a)for average communication - 16; b)for coefficient of variation - 10.

Table 2: Relation Between Performance and The Variation (st.dev/mean) Across Project Members' Communication Within the Project

Results from Allen, Lee and Tushman (1980)[1]	Results from the Software Environment (present study)[1] Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research -0.50**	0.00	0.04	0.00	0.00	0.08
Development -0.01	---[Most similar to present study]				
Technical Service 0.28					

1-Kendall's Tau. ** p=0.05. See footnote for the meaning of the coefficient of variation.

Though positive relations between intra-project communication and performance might be expected, there are several possible causes for the above results. First, the project teams are not larger than eleven members, which makes their coordinating requirements, in view of Brooks's premises, minimal. Second, because the number of team members vary between 6 and 11, the coordinating requirements do not vary significantly across projects. In addition, we might assume that the project members expertise is based on similar core technologies, which should reduce the value of innovation oriented internal communication.

Communication within the department

The department usually consists of thirty people, all of them technical. The communication intensity within the department clearly contributes to project performance on all of the performance criteria, statistically significant for meeting designated schedule, functional specifications and overall project success (Table 3). In a sense the contribution is valuable for both the coordination oriented and the innovation oriented criteria.

Table 3: Relation Between Performance and Communication Among Department Members

Results from the Software Environment (present study)[1] Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
0.13	0.37**	0.40**	0.22	0.28*

1-Kendall's Tau. ** p=0.05, * p=0.10

The data in Table 4 suggest that the intradepartmental communication should be evenly spread among the project members, especially if consider the functionality and reliability criteria.

Coefficient of variation (standard deviation divided by the mean) can be used as a measure of the degree to which a few individuals monopolize communication. A coefficient close to zero indicates a uniform distribution of communication among project members; a high coefficient indicates significant monopolization of communication by the few.

Table 4: Relation Between Performance and the Variation (st.dev/mean) Across Project Members' Communication Within the Department

Results from the Software Environment (present study)[1]				
Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
-0.12	-0.22	-0.36**	-0.26*	-0.22

1-Kendall's Tau. ** p=0.05, * p=0.10

Although there is no analogue for this intermediate grouping in Allen et al., the results are quite similar to the next level of the organizational structure - the Business Center/Sector or the division in Allen et al. These data are presented in Tables 5 and 6.

Communication with the rest of Applied Systems

As mentioned before, a Business Center/Sector is addressing a specific market segment but is not based on a specific technology. In addition to technical staff it includes marketing experts as well. It is significantly larger than a department, between 80 and 200 people strong. The relations between communication with people within the Business Center/Sector and project performance in the software environment (Table 5) resemble the relations at the technical service setting, in Allen et al. study. The presumably statistically significant correlation with functionality is rendered insignificant in view of a higher probability for its occurance solely by chance due to multiple comparisons.

Table 5: Relation Between Performance and Communication With the Rest of the Division: Intra-Business Center

		Results from the Software Environment (present study)[1]				
Allen, Lee and Tushman (1980)[2]		Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research	-0.17	0.13	0.20	0.30*	0.23	0.16
Development	0.31*					
Technical Service	0.18	---[Most similar to present study]				

1-Kendall's Tau. 2-Pearson Correlation. * p=0.10

The relations between communication variation and performance are also similar to those of technical service in the sense that the distribution of communication of the software development project team with the divisional staff, external to its immediate department, is inconsequential for proejct performance (Table 6). This is quite different from Allen et al. results for development projects, which benefited from evenly spread communications.

The similarity of the results for software development with technical service continues for the inter Business Center/Sector communications (Table 7 and 8). It is reasonable to assume that the coordinating value of these communication is non-existent, while the technology carrying communication is either irrelevant or missing. The negative relations are somewhat puzzling.

Table 6: Relation Between Performance and The Variation (st.dev/mean) Across Project Members' Communication with the Rest of of the Division: Intra-Business Center

		Results from the Software Environment (present study)[1]				
Results from Allen, Lee and Tushman (1980)[1]		Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research	-0.17	-0.03	-0.02	-0.09	-0.02	0.00
Development	-0.25*					
Technical Service	-0.02	---[Most similar to present study]				

1 Kendall's Tau. * p=0.10

They might be the result of a situation when intensive communication was spurred by an internal problem, which this communication could not resolve.

Table 7: Relation Between Performance and Communication With the Rest of the Division: Inter-Business Center

		Results from the Software Environment (present study)[1]				
Results from Allen, Lee and Tushman (1980)[2]		Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research	-0.34	-0.44**	-0.30*	-0.35**	-0.47**	-0.44**
Development	0.09					
Technical Service	-0.47**	---[Most similar to present study]				

1-Kendall's Tau. 2-Pearson Correlation. ** p=0.05, * p=0.10

Table 8: Relation Between Performance and The Variation (st.dev/mean) Across Project Members' Communication with the Rest of of the Division: Inter-Business Center

		Results from the Software Environment (present study)[1]				
Allen, Lee and Tushman (1980)[1]		Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Research	-0.20	0.28	0.24	0.34*	0.49**	0.51**
Development	-0.27**					
Technical Service	0.15	---[Most similar to present study]				

1 Kendall's Tau. ** p=0.05, * p=0.10.

The most relevant network: programmers, designers or managers

The various communication networks between the project team and the rest of the division were separated into programmers, designers and managers. Although most of the managers, at least at the level of project and department managers, are technically trained, still, in comparison with purely technical staff such as designers and programmers, they represent the formal, coordinative dimension of the organization.

The intra-departmental data in Table 9 indicates that its managers, the department head, project managers, and team leaders provide the most useful information to the project. Although this information can still be innovation oriented, in view of the positional responsibilities of managers, it should be coordinative as well. Those managers might be the departmental gatekeepers, though it should be noted that in contrast with Allen & Cohen (1970) description of gatekeepers (see also Allen, 1977) they are usually second or third level supervisors, and they do not design or program themselves.

Table 9: Relation Between Performance and Communication within the Department with Programmers, Designers and Managers

Communication with:	Results from the Software Environment (present study)[1] Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Programmers	-0.08	0.17	0.23	0.03	0.14
Designers	0.02	0.11	0.16	0.04	0.08
Managers	0.21	0.36**	0.39**	0.30*	0.37**

1 Kendall's Tau. ** p=0.05, * p=0.10.

On the other hand, the communication with neither programmers, designers or managers should not be channelled through few individuals: the lower is the coefficient of variation - the higher is the project performance (Table 10). It indicates that when the departmental management team maintains open and direct channels of communication with most of the project team members, it succeeds to provide them with the necessary information.

Table 10: Relation Between Performance and The Variation (st.dev/mean) Across Project Members' Communication within the Department with Programmers, Designers and Managers

Communication with	Results from the Software Environment (present study)[1] Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Programmers	-0.11	-0.27*	-0.33**	-0.17	-0.20
Designers	-0.28	-0.44*	-0.44*	-0.54**	-0.50**
Managers	-0.18	-0.29*	-0.42**	-0.19	-0.26*

1 Kendall's Tau. ** p=0.05, * p=0.10.

Similar analysis for communication external to the department and the Business Center/Sector yields statistically insignificant results for all three networks. It is not surprising in view of the data in Tables 5 and 7 above, though they re-emphasize the local nature of the software development project environment.

Exposure to additional sources of information

The previous results suggest a very narrow focus of the software development activities. The farther we move from the department, the less useful is the information the that environment can offer. It seems that beyond the immediate domain of the department there are no sources of information which might be of any use to the project team.

Table 11: Relation Between Performance and Use of Various Sources of Information: Literature, Conferences and Outsiders

Sources of information	Results from the Software Environment (present study)[1] Meeting Designated Budget	Meeting Designated Schedule	Meeting Functionality Specifications	Success in Reliability	Overall Project Success
Literature					
Informal	0.37**	0.30*	0.33*	0.21	0.31*
Formal	-0.13	-0.19	-0.15	-0.30*	-0.17
Conferences	0.01	-0.02	-0.06	0.04	0.16
External	-0.06	-0.14	-0.08	-0.06	-0.15

1 Kendall's Tau. ** p=0.05, * p=0.10.

The data in Table 11 re-enforce this perception to great extent. Only the informal literature, which consists mostly of internal Applied Systems and ICL reports, software and hardware manuals, with insignificant amount of really external publications, is useful to the project team. All the other sources of information, all external, do not provide a positive input.

Conclusions

Software development as R&D and manufacturing

Both literature and data presented above suggest that software development might not be what it seems. Recent literature actually introduced a more specific and appropriate vocabulary to describe the relevant activities - software engineering and software production (Somerville, 1982; Freeman, 1980; Frank, 1983). A new notion is also that software development is not only programming; there are many additional activities such as design, maintenance, validation, documentation, and many others.

The pre-occupation with productivity is usually much more typical of the manufacturing function in organizations. On the other hand, R&D managers are usually interested in innovativeness, creativity, and technical performance of their staff. The promotion strategies of hardware and software products for the software development process are indicative of this concern with productivity. Most of them claim that their products either "Trim software development time by 75%", "Save in development 54%, maintenance 75%", or "Boost programming productivity up to 75%" (Frank, 1983: 64-65). Somerville's (1980) discussion of programming practices focuses on tools such as interactive computer technologies, structured programming, and languages. Frank's comprehensive review of the industry (1983) emphasizes the cases of the MARK IV high-level computer language, and ADA - a programming technique. All those are purely process technologies.

My informal communications with software development managers at Applied Systems, ICL validate this situation. Some of them indicated that they had already started to describe application programming as "software fabrication" or "code-cutting". On the other hand, most emphasized that the development of system and super-structure software is more complex, unstructured, and technologically volatile. It presents a challenge different from the "code-cutting" oriented activities, more typical of application programming. Development of this type of products will resemble R&D more than manufacturing, in the hardware vocabulary.

In view of this information, the overwhelming similarity of the relations between communication and performance with technical service should be

treated with caution. On the one hand, judging by the insignificant contribution of communication external to the project team, especially intra-firm communication, software development does not qualify as a development activity, in the R&D spectrum. On the other hand, because the assortment of the projects in the sample mostly includes application software, with only a few super-structures, and probably a single system program, the above results might be valid only for this type of software products and tasks. These results are still preliminary; additional data available from the Applied Systems should be used to validate them. A study of an additional software environment, containing a sufficient number of projects, representative of operating systems, super-structures, and applications, will be very useful for internal and external validity.

Still, on the basis of the data ad hoc, "software development" is a misnomer; more specific constructs of R&D, production, and probably maintenance should be used for its composists. But more than that - these composists should be managed differently. While the coordinative approach might be appropriate for the quasi-manufacturing, and the maintenance activities, the premises of communication as carrier of technological innovation will apply to software R&D. It implies that the image of software development being so different from hardware R&D, in the context of information flow, is only true for the quasi-manufacturing activities. Software engineering and design probably will benefit from informal communication, which will be the familiar vehicle of technological innovation, similarly to non-software R&D. This hypothesis could and should be tested through additional rigorous research.

The coordination-innovation trade-off

The results from software development, in addition to Allen et al. (1980) data from technical services emphasize an interesting point. First, numerous studies based on the premises of communication-as-innovation-carrier showed that internal communication of project team members with other professionals in the R&D laboratory is a strong determinant of project performance. On the other hand, the monopolization of communication with these professionals by project managers was beneficial for technical services.

These diagonally opposite views can be integrated when we regard innovation and coordination requirements in a trade-off situation. Coordination oriented communication could be more salient than innovation carrying communication for tasks which are more structured, based on a stable, maturing technology. These are tasks at the lower end of the R&D spectrum, represented by technical services, and probably software production. They do not require a wide variety of technical expertise for their optimal performance. They also are not as acutely threatened, as many R&D departments of technology-based organizations, especially those operating in a project structure, by the rapid professional obsolescence of their technical staff.

An alternative perspective for this phenomenon could be the interdepedence level of task activities. If the task composites require close interaction with each other, the coordination requirements will increase parabolically. Although technical services do not fit this mold, software projects seem to present such a highly interdependent environment. Because the interdependence of software program's modules is inherent, a closely controlled interaction is a strong pre-requisite of adequate functionality, usability, reliability, maintainability, and flexibility.

In view of Allen et al. results from technical services, it is clear that the issue of the coordination-innovation trade-off presents itself in hardware R&D. The fact that it had not been addressed as yet is the result of the somewhat one-sided focus of R&D management researchers on technological

innovation. The analysis of this trade-off, and its significant potential implications for organizational design of R&D, suggests an exciting agenda for future research in the area of R&D management.

References

Abdel-Hamid, T. K. (1984). The dynamics of software development project management: An integrative System Dynamics perspective. Unpublished doctoral dissertation, MIT. Cambridge, Massachusetts.

Allen, T. J. (1977). Managing the flow of technology. Cambridge, Massachusetts: MIT Press.

Allen, T. J., & Cohen, S. (1969). Information flow in R&D laboratories. ASQ, 14, 12-19.

Allen, T. J., Tushman, M. L., & Lee, D. M. S. (1979). Technology transfer as a function in the spectrum from research through development to technical services. Academy of Management Journal, 22(4), 694-708.

Allen, T. J., Lee, D. M. S., & Tushman, M. L. (1980). R&D perfromance as a function of internal communication, project management, and the nature of work. IEEE Transactions on Engineering Management, EM-27(1), 2-12.

Boehm, B. W. (1980). Software engineering - as it is. In H. Freeman & P. M. Lewis III (eds.). Software engineering (pp.37-73). New York: Academic Press.

Brooks, F. P. Jr. (1982). The mythical man-month (2nd ed.). Reading Mass: Addison-Wesley.

Dijkstra, E. W. (1976). A discipline of programming. Englewood Cliffs, New Jersey: Prentice-Hall.

Frank, L. F. (1983). Critical issues in software. New York: Wiley.

Freeman, P. (1980). The central role of design in software engineering: Implications for research. In H. Freeman & P. M. Lewis III (eds.). Software engineering (pp.121-132). New York: Academic Press.

Howlett, J. (1985). Informal correspondance. Harwell.

Libby, R., & Blashfield, R. K. (1978). Performance of a composite as a function of the number of judges. Organizational Behavior and Human Performance, 21, 121-129.

Mills, H. D. (1976). Software development. IEEE Transactions on Software Engineering, 68(9).

Pelz, D. C., & Andrews, F. M. (1966). Scientists in organizations: Productive climates for R&D. New York: Wiley.

Scott, R. F., & Simmons, P. B. (1975). Predicting programming group productivity - a communication model. IEEE Transactions on Software Engineering, SE-1(4).

Somerville, I. (1982). Software engineering. London: Addison-Wesley.

bar code is on back cover